EXHIBIT C-3
EXEMPLARY PORTIONS OF PRIOR ART THAT TEACH OR SUGGEST EACH ELEMENT OF THE ASSERTED '661 CLAIMS
PATENT L.R. 3-3(C)

Claim 1 ('661 Patent)	**Sci.crypt Postings (1995)**
A cryptographic processing device for securely performing a cryptographic processing operation including a sequence of instructions in a manner resistant to discovery of a secret by external monitoring, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program." Jim Bell (12/24/95) – "Okay, I admit, this is certainly not a new idea. The military's TEMPEST program is to build electronic equipment which is so 'quiet' that it is impossible (or, at least, arbitrarily difficult) to capture useful information by inadvertent radio transmission."
(a) an input interface for receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it

	ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program." Input interface is inherent in cryptographic device. Input interface is inferred from discussion of input data and network.
(b) a source of unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) a processor:	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Processor can be inferred from performance of operations.
(i) connected to said input interface for receiving and cryptographically processing said quantity,	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" A connection between the processor and the input interface is inherent.
(ii) configured to use said unpredictable information to	Jim Bell (12/24/95) – "[T]he 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring

conceal a correlation between externally monitorable signals and said secret during said processing of said quantity by modifying said sequence; and	of the computer can be done (it will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(d) an output interface for outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program." An output interface is inherent.

Claim 2 ('661 Patent)	**Sci.crypt Postings (1995)**
The device of claim 1 wherein said input interface and said output interface are the	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the

same element.	overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." *See also, e.g.*, U. S. Patent No. 5,068,894 to Hoppe at, *e.g.*, 5:4-29; John F. Wakerly, MICROCOMPUTER ARCHITECTURE AND PROGRAMMING; THE 68000 FAMILY, John Wiley & Sons, Inc., New York, 1997 at 6-7.

Claim 4 ('661 Patent)	**Sci.crypt Postings (1995)**
The device of claim 1 wherein said cryptographic processing operation includes transforming a message with the Data Encryption Standard (DES).	Mutatis Mutandis (12/12/95) – "Any more work with symmetric algorithms or hash functions? DES comes to mind: especially if 'approved' (?) implementations are in hardware only." *See also, e.g.*, Menezes, A.J. et al., HANDBOOK OF APPLIED CRYPTOGRAPHY, CRC Press, Boca Raton at 223 and 250 (1997); "Data Encryption Standard," Federal Information Processing Standards Publication (FIPS PUB) 46-2, U.S. Department of Commerce, National Institute of Standards and Technology, Dec. 30, 1993 (suggesting, at 3, implementing DES with microprocessors using ROM, PROM or EEROM; and describing, at 5, high level of protection provided by DES).

Claim 5 ('661 Patent)	**Sci.crypt Postings (1995)**
A cryptographic processing device for securely performing a cryptographic processing operation implementing a permutation in a manner resistant to discovery of a secret by external monitoring, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "A second way to defeat Kocher's attack is to

	use blinding: you 'blind' the data beforehand, perform the cryptographic computation, and then unblind afterwards. For RSA, this is quite simple to do. (The blinding and unblinding operations still need to take a fixed amount of time.) This doesn't give a fixed overall computation time, but the computation time is then a random variable that is independent of the operands." Bill Stewart (12/13/95) – "The timing attack on Diffie-Hellman depends on assumptions about what multiplications are being made, and in what order. But you don't need to do them in order." Bill Stewart (12/13/95) – "The standard approach to calculating Y**x mod m is to calculate Y[1]=Y, Y[2] = Y**2, Y[3]=Y**4, ... Y[log x] = Y**(log x), and while you're doing this keep a running total r[i], where r[i] = (bit[x,i]) ? (r[i-1]*Y[i]) : r[i-1] all arithmetic modulo m ... This may be a bit memory-intensive for a smartcard, but there's no need to calculate these partial products in order; precompute the Y[i], pick a random permutation of 1..(log x), and compute the partial products in that order." Bill Stewart (12/13/95) – "As a further annoyance to the listener, split the permutation at random into two or three pieces, compute their products separately, and then multiply those partial products together."
(a) an input interface for receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(b) a source of unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) a processor:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a

	decryption takes in a computer of a known cpu speed."
(i) connected to said input interface for receiving and cryptographically processing said quantity,	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(ii) configured to use said unpredictable information to conceal a correlation between externally monitorable signals and said secret during said processing of said quantity by randomizing the order of said permutation; and	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "A second way to defeat Kocher's attack is to use blinding: you 'blind' the data beforehand, perform the cryptographic computation, and then unblind afterwards. For RSA, this is quite simple to do. (The blinding and unblinding operations still need to take a fixed amount of time.) This doesn't give a fixed overall computation time, but the computation time is then a random variable that is independent of the operands." Bill Stewart (12/13/95) – "The timing attack on Diffie-Hellman depends on assumptions about what multiplications are being made, and in what order. But you don't need to do them in order." Bill Stewart (12/13/95) – "The standard approach to calculating Y**x mod m is to calculate Y[1]=Y, Y[2] = Y**2, Y[3]=Y**4, ... Y[log x] = Y**(log x), and while you're doing this keep a running total r[i], where r[i] = (bit[x,i]) ? (r[i-1]*Y[i]) : r[i-1] all arithmetic modulo m ... This may be a bit memory-intensive for a smartcard, but there's no need to calculate these partial products in order; precompute the Y[i], pick a random permutation of 1..(log x), and compute the partial products in that order." Bill Stewart (12/13/95) – "As a further annoyance to the listener, split the permutation at random into two or three pieces, compute their products separately, and then multiply those partial products together."

(d) an output interface for outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."

Claim 6 ('661 Patent)	**Sci.crypt Postings (1995)**
A cryptographic processing device implemented on a single microchip for securely performing a cryptographic processing operation in a manner resistant to discovery of a secret by external monitoring, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least. Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program. Jim Bell (12/24/95) – "Okay, I admit, this is certainly not a new idea. The military's TEMPEST program is to build electronic equipment which is so 'quiet' that it is impossible (or, at least, arbitrarily difficult) to capture useful information by inadvertent radio transmission." *See also, e.g.*, Scott Guthery, "Smart Cards," May 28, 1998, www.usenix.org/publications/login/1998-5/guthery.html (visited Dec. 5, 2006) ("Single-chip smart card processors based on these cores are made by almost all the large silicon foundries . . . .Several marketplace forces are at work to open the smart card as a general-purpose computing platform.").

(a) an input interface for receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program."
(b) a source of unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) a processor:	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(i) connected to said input interface for receiving and cryptographically processing said quantity,	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt

	generator might help, here.)"
(ii) configured to use said unpredictable information to conceal a correlation between said microchip's power consumption and said processing of said quantity by expending additional electricity in said microchip during said processing; and	Jim Bell (12/24/95) – "[T]he 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is [sic] to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(d) an output interface for outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program."

Claim 7 ('661 Patent)	**Sci.crypt Postings (1995)**
The device of claim 6	Jim Bell (12/24/95) – "[T]he 'solution' that is typically discussed

including program logic to activate said expending during said processing.	involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is [sic] to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"

Claim 11 ('661 Patent)	**Sci.crypt Postings (1995)**
A cryptographic processing device for securely performing a cryptographic processing operation in a manner resistant to discovery of a secret by external measurement of said device's power consumption, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program." Jim Bell (12/24/95) – "Okay, I admit, this is certainly not a new idea. The military's TEMPEST program is to build electronic equipment which is so 'quiet' that it is impossible (or, at least, arbitrarily difficult) to capture useful information by inadvertent radio

	transmission." Jim Bell (12/24/95) – "More than a year ago, it occurred to me that it might be worth it to build a CPU clock replacement module that took the place of the main CPU crystal oscillator, and replaced it with a oscillator module whose frequency was (very long period!) pseudorandomly varied, possibly with a resolution of 16-bit and over a range of perhaps 1%., with the frequency varying every few tens or hundreds of microseconds. The result, I presume, is that every operation synchronized to the microprocessor clock would vary in time and would be hard to 'tune' with a normal radio receiver. It seems to me that this would make the resulting computer harder to 'bug' using standard equipment." Jim Bell (12/24/95) – "If this were do-able and were in fact done, it would probably be worth it to 'tailor' the spectrum of variation in clock speed so that these variations do not tend to average out over 'long' times, for example a few hundred milliseconds or even tens of seconds. This would at least help to disguise the decryption-time information that is commonly discussed."
(a) an input interface for receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program."
(b) an input interface for receiving a variable amount of power, said power consumption varying	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the

measurably during said performance of said operation;	overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(c) a processor connected to said input interface for receiving and cryptographically processing said quantity; and	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(d) a noise production system for introducing noise into said measurement of said power consumption.	Jim Bell (12/24/95) – "A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end.." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is [sic] to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "The simplest way to defeat Kocher's timing attack is to ensure that the cryptographic computations take an amount of time that does not depend on the data being operated on. For example, for RSA it suffices to ensure that a modular multiplication always takes the same amount of time, independent of the operands."

Claim 22 ('661 Patent)	**Sci.crypt Postings (1995)**
A device according to claims 1, 4, 7, 9, 11, 14, 15, or 20 wherein said device comprises a smartcard.	Markus Kuhn (12/12/95) – "Who is talking only about networks? RSA, DH, DSS, etc. are widely used in smart cards. For smart cards, you even have to provide the CPU clock signal, so you can count the number of bus clock cycles required to do the calculation." Bill Stewart (12/13/95) – "You don't have to be particularly intrusive to get timing information from a smartcard plugged into your smartcard reader, such as a digital wallet or Nationalized ID Card. Multiple precision arithmetic is slow, especially for things like

	exponentiation on 1000-bit numbers and smartcards are already slow. If you've got a smartcard plugged into your own machine, it's possible to get a certain amount of timing information from it during transactions across a network."

Claim 23 ('661 Patent)	Sci.crypt Postings (1995)
A method of securely performing a cryptographic processing operation in a manner resistant to discovery of a secret within a cryptographic processing device by external monitoring, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed."
(a) receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(b) generating unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) cryptographically processing said quantity, including using said unpredictable information while processing said quantity to conceal a correlation	Jim Bell (12/24/95) – "It occurs to me that softrware [sic] should be written to complete operations in an identical time frame, no matter the input data. While this has already been hashed over on the nets, the 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will

between externally monitorable signals and said secret by selecting between:	know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end."
(c)(1) performing a computation and incorporating the result of said computation in said cryptographic processing, and	Jim Bell (12/24/95) – "It occurs to me that softrware [sic] should be written to complete operations in an identical time frame, no matter the input data. While this has already been hashed over on the nets, the 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end."
(c)(2) performing a computation whose output is not incorporated in said cryptographic processing; and	Jim Bell (12/24/95) – "It occurs to me that softrware [sic] should be written to complete operations in an identical time frame, no matter the input data. While this has already been hashed over on the nets, the 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] only after the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end."
(d) outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."

Claim 24 ('661 Patent)	**Sci.crypt Postings (1995)**
The method of claim 23 where said selecting is	Jim Bell (12/24/95) – "It occurs to me that softrware [sic] should be written to complete operations in an identical time frame, no matter the input data. While this has already been hashed over on the nets,

performed in software.	the 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] onlyafter the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end."

Claim 25 ('661 Patent)	**Sci.crypt Postings (1995)**
The method of claim 23 where said selecting is performed in hardware on an integrated circuit including a microprocessor.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed." Jim Bell (12/24/95) – "It occurs to me that softrware [sic] should be written to complete operations in an identical time frame, no matter the input data. While this has already been hashed over on the nets, the 'solution' that is typically discussed involves adding a null loop at the end of the real operation, and contining [sic] onlyafter the 'wall clock' shows enough time has passed. This isn't an adequate solution, I think, if local RFmonitoring of the computer can be done. (It will know when the actual result ended) A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end."

Claim 27 ('661 Patent)	**Sci.crypt Postings (1995)**
A method of securely performing a cryptographic processing operation including a sequence of instructions in a manner resistant to discovery of a secret within a cryptographic processing device by external monitoring,	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with

comprising:	pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(a) receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(b) generating unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) using said unpredictable information while processing said quantity to conceal a correlation between externally monitorable signals and said secret by using said unpredictable information to modify said sequence; and	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(d) outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."

Claim 28 ('661 Patent)	**Sci.crypt Postings (1995)**
A method of securely performing a cryptographic processing operation implementing a permutation in a manner resistant to discovery of a secret within a cryptographic processing device by external monitoring, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "A second way to defeat Kocher's attack is to use blinding: you 'blind' the data beforehand, perform the cryptographic computation, and then unblind afterwards. For RSA, this is quite simple to do. (The blinding and unblinding operations still need to take a fixed amount of time.) This doesn't give a fixed overall computation time, but the computation time is then a random variable that is independent of the operands." Bill Stewart (12/13/95) – "The timing attack on Diffie-Hellman depends on assumptions about what multiplications are being made, and in what order. But you don't need to do them in order." Bill Stewart (12/13/95) – "The standard approach to calculating Y**x mod m is to calculate Y[1]=Y, Y[2] = Y**2, Y[3]=Y**4, ... Y[log x] = Y**(log x), and while you're doing this keep a running total r[i], where r[i] = (bit[x,i]) ? (r[i-1]*Y[i]) : r[i-1] all arithmetic modulo m ... This may be a bit memory-intensive for a smartcard, but there's no need to calculate these partial products in order; precompute the Y[i], pick a random permutation of 1..(log x), and compute the partial products in that order." Bill Stewart (12/13/95) – "As a further annoyance to the listener, split the permutation at random into two or three pieces, compute their products separately, and then multiply those partial products together."
(a) receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the

message;	overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(b) generating unpredictable information;	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)"
(c) using said unpredictable information while processing said quantity to conceal a correlation between externally monitorable signals and said secret by randomizing the order of said permutation; and	Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudorandom order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "A second way to defeat Kocher's attack is to use blinding: you 'blind' the data beforehand, perform the cryptographic computation, and then unblind afterwards. For RSA, this is quite simple to do. (The blinding and unblinding operations still need to take a fixed amount of time.) This doesn't give a fixed overall computation time, but the computation time is then a random variable that is independent of the operands." Bill Stewart (12/13/95) – "The timing attack on Diffie-Hellman depends on assumptions about what multiplications are being made, and in what order. But you don't need to do them in order." Bill Stewart (12/13/95) – "The standard approach to calculating Y**x mod m is to calculate Y[1]=Y, Y[2] = Y**2, Y[3]=Y**4, ... Y[log x] = Y**(log x), and while you're doing this keep a running total r[i], where r[i] = (bit[x,i]) ? (r[i-1]*Y[i]) : r[i-1] all arithmetic modulo m ... This may be a bit memory-intensive for a smartcard, but there's no need to calculate these partial products in order; precompute the Y[i], pick a random permutation of 1..(log x), and compute the partial products in that order." Bill Stewart (12/13/95) – "As a further annoyance to the listener, split the permutation at random into two or three pieces, compute their products separately, and then multiply those partial products

	together."
(d) outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."

Claim 29 ('661 Patent)	**Sci.crypt Postings (1995)**
A method of securely performing a cryptographic processing operation in a manner resistant to discovery of a secret within a cryptographic processing device by external monitoring of said device's power consumption, comprising:	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program." Jim Bell (12/24/95) – "Okay, I admit, this is certainly not a new idea. The military's TEMPEST program is to build electronic equipment which is so 'quiet' that it is impossible (or, at least, arbitrarily difficult) to capture useful information by inadvertent radio transmission." Jim Bell (12/24/95) – "More than a year ago, it occurred to me that it might be worth it to build a CPU clock replacement module that took the place of the main CPU crystal oscillator, and replaced it with a

	oscillator module whose frequency was (very long period!) pseudorandomly varied, possibly with a resolution of 16-bit and over a range of perhaps 1%., with the frequency varying every few tens or hundreds of microseconds. The result, I presume, is that every operation synchronized to the microprocessor clock would vary in time and would be hard to 'tune' with a normal radio receiver. It seems to me that this would make the resulting computer harder to 'bug' using standard equipment." Jim Bell (12/24/95) – "If this were do-able and were in fact done, it would probably be worth it to 'tailor' the spectrum of variation in clock speed so that these variations do not tend to average out over 'long' times, for example a few hundred milliseconds or even tens of seconds. This would at least help to disguise the decryption-time information that is commonly discussed."
(a) receiving a variable amount of power, said power consumption varying measurably during said performance of said operation;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."
(b) receiving a quantity to be cryptographically processed, said quantity being representative of at least a portion of a message;	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least." Jim Bell (12/24/95) – "However, being a ham and occasionally listening to the various odd noises produced by a computer when you tune a VHF or UHF ham radio to a harmonic of the clock speed, it ccurred [sic] to me that the delay times WITHIN a particular encryption/decryption would be far more easily measured with local RF snooping. I would imagine that if you can determine even a fraction of a bit of key from a network 'ping,' you could do a lot better 'listening' to the execution of a program within a few hundred feet with an ordinary radio receiver and a sophisiticated [sic] analysis program."

(c) introducing noise into said measurement of said power consumption while processing said quantity; and	Jim Bell (12/24/95) – "A better (and, sadly, more inefficient) method would involve executing BOTH branches of conditional jump, and only using the data generated from the desired half at the very end.." Jim Bell (12/24/95) – "Another possibility might be (for certain large mathematical algorithms) is [sic] to split up the functions and to execute them in a 'random' order, with enough 'dummy' operations inserted to further disguise the facts. For example, if you're multiplying two 1024-bit values to get a 2048-bit result, program this to be done in a pseudrandom [sic] order and intersperse any operations with pseudorandom operations to disguise it. (A pseudorandom interrupt generator might help, here.)" Ron Rivest (12/11/95) – "The simplest way to defeat Kocher's timing attack is to ensure that the cryptographic computations take an amount of time that does not depend on the data being operated on. For example, for RSA it suffices to ensure that a modular multiplication always takes the same amount of time, independent of the operands."
(d) outputting said cryptographically processed quantity to a recipient thereof.	Jim Bell (12/24/95) – "Recently there has been a substantial amount of discussion concerning the use of accurate timing in an attempt to uncover encryption keys, by carefully noting the length of time that a decryption takes in a computer of a known cpu speed. I have noticed that most of the discussion focussed [sic] on the delay time of the overall operation done over a network, for example a LAN or perhaps the Internet, but it has been recognized that imprecision due to indeterminate network timings make such a tactic problematic at least."